アドベンチャーゲーム制作講座

㈱マイクロキャビン
橋本 浩幸

アドベンチャーゲームは、あなたもつくれる

近頃、パソコン用のゲームとして、アドベンチャーゲームが話題を集めています。アドベンチャーゲームは、もともと日本には無かったゲームで、アメリカが原産地です。したがって、日本ではまだその種類も少なく、それについて書かれた解説書もないのが現状です。

一見、自分でつくるのはとても大変なようですが、はたから考える程、アドベンチャーゲームをつくるには、特別なテクニックを必要としません。マイコンホビーストにとって、願ってもない世界だと言えるでしょう。

ここでは、「アドベンチャーゲームというものの紹介を兼ねて、**〈これからアドベンチャーゲームを作りたいと考えている人の為に〉** この「不思議の国のアリス」が、どの様に作られているかを、紹介したいと思います。(尚、この文章は、㈱新紀元社刊「PC-マガジン」に連載されたアドベンチャーゲーム制作講座を一冊にまとめ上げたものです。この企画を承知してくださった新紀元社編集部に感謝いたします。)

●アドベンチャーゲームは「謎解き」ゲームだ。

アドベンチャーゲームとは、一種の謎解きゲームで、画面に書かれた文章や絵の中からヒントを見つけ、さまざまな謎を解き、所定の目的を達成するというゲームです。

プレイヤーは、パソコンがつくり出した架空の家の中や、街や、宇宙空間などをさまよい、手がかりを見つけ出していきます。パソコンは、その地点の情景や状況などを出力しながら、プレイヤーの動作（動く、捜す、見るなど）を受け付けていきます。

プレイヤーは、ドアにカギがかかっている場所では、カギをはずし、崖があればそれを登り、宇宙に出る時には、宇宙船を手にいれなければなりません。こうして、何回もトライアンドエラーを繰り返しながら、ゲームは進んでいきます。

このように、アドベンチャーゲームとは、一種の会話型ゲームと言えるでしょう。したがって、様々な入力に対して、コンピュータが如何にうまく応答するかも、ゲームの面白さを決める重要な要素になってきます。ゲームを進めながら、あたかも自分が冒険世界に入り込んだような気になれるのが、アドベンチャーゲームの醍醐味です。

アドベンチャーゲームには、画面の出力が文章だけのものと、文章と絵が同時に出力されるものとがあります。どちらも本質的には同じで、絵では表現が困難なもの（動作などを多く扱う場合は前者となり、文章では冗長になってしまう場合は後者となります。

●アドベンチャーゲームにもいろいろある。

一口にアドベンチャーゲームといっても，その中には、さまざまなジャンルのものがあります。例えば、ゲームの舞台もいろいろあり、部屋の中や街の中、宇宙空間や時間旅行ミステリー、夢の中、魔法世界などのファンタスティックなものまで様々です。

これらの舞台によって、アドベンチャーの内容も変化してきます。例えば、部屋の中が舞台になる時、何かあるものを捜す、ということが、そのアドベンチャーゲームの目的、手段に多く用いられます。ですから、このような舞台では、「何があるか」ということが重要な要素になります。

これに対して、街の中や宇宙空間、時間旅行などでは、プレイヤーの動作が、そのままゲームの内容に結びつきますので、「何をするか」が重要になります。また、特に夢の中や魔法世界では、現実では不可能な動作が可能なため、「何ができるか」が重要になります。

この、「何があるか」と「何をするか」「何が

できるか」との違いは、ゲーム内容に特に大きく影響を及ぼします。「何があるか」を問題にする場合、プレイヤーの入力は、「見る」「捜す」「取る」など、限定された単語だけで済みます。プレイヤーは、どこでその単語を使うか考えればよいわけで、使える単語の一覧表（本当は、これは邪道なのですが）をつくることもできます。

これに対して、「何をするか」「何ができるか」が問題になる場合、各場面に対する入力は、それぞれ全く異なります。このためプレイヤーは、あらゆる可能性の中から、最もその場面に適した動作を考えて入力しなくてはなりません。

これらは、もちろん、プログラミングにも大きく影響してきます。

また、ゲームの最終目的にも様々なものがあります。一般的には、何かを捜して見つけ出すということが、最終目的として多く用いられているようです。何かを捜して、それを使うという場面は、どんなジャンルのアドベンチャーゲームにも、一度は出てくるものですし、これだけで構成されたアドベンチャーゲームも沢山あります。「捜し物」はアドベンチャーゲームの基本であるとも言えるでしょう。

「犯人捜し」などの謎解きも、アドベンチャーゲームの目的として多く使われます。これは、推理小説をそのままプログラムにすればよいわけで、伏線も張りやすく、アドベンチャー向きの題材というわけです。

しかし、プログラマーの創意工夫によっては、何でも題材になり得るわけで、既成の概念にとらわれずにつくられたアドベンチャーゲームが（最近では、ガールハントまでアドベンチャーゲームになっています!?）、最も面白いようです。このあたりは、小説を書くことにも似ています。

●アドベンチャーゲームと他のゲームはどう違うのか？

前述したように、アドベンチャーゲームは、従来のパソコン用ゲームとは全く異なるタイプのゲームです。特に反射神経型のリアルタイムゲームとは対照的な違いがあります。

リアルタイムゲームには、目的とするものはありません。その代わりになるものとして「得点」があります。プレイヤーが様々な状況を失敗せずにクリアーし、得点を稼いでいくことが、リアルタイムゲームの面白さになります。

表1 アドベンチャーゲームと他のゲームとの比較

	アドベンチャーゲーム	リアルタイムゲーム	ウォーゲーム	ロールプレイングゲーム
最終目的	あり（宝捜し、犯人捜しなど）	なし	あり（敵のせん滅など）	なし（あるものもある）
得点	なし	重要なウェイトを占める	なし	なし
ゲーム経過	ただ一通りを試行錯誤する	毎回異なった過程をもつ	ほぼ決まった（定石）過程をもつ	毎回異なった過程をもつ
ゲーム終了	目的達成時	（味方宇宙船など）破壊時など	目的達成時	キャラクター死亡時など

これに対して、アドベンチャーゲームには「得点」はなく、代わりに確固とした目的を持っています。プレイヤーが何回も失敗を繰り返して、所定の目的を達成することが、アドベンチャーゲームの面白さになります。したがって、リアルタイムゲームの面白さと、アドベンチャーゲームの面白さは、全く異質なものと言えるでしょう。

また、アドベンチャーゲームと同じく、最近注目されているゲームにシミュレーションゲームがあります。このゲームは、今まで無かったジャンルというわけではなく、「スタートレック」という、コンピュータゲームとしては最古のものから将棋などに至るまで、シミュレーションゲームの一種と考えられます。このようにシミュレーションゲームは、頭脳型のゲームとしては最高の部類に入るもので、アドベンチャーゲームとの違いは、目的を達成するまでの過程が何通りもあり、最も良い状況をつくり出すことがゲームの主眼点です。このため、「定石」が存在するものも、このゲームの特徴です。

これに対して、アドベンチャーゲームが進行していく筋道は、ただ一つしかありません。プレイヤーは、その道を求めて、試行錯誤を繰り返していくわけです。

さらに、ロールプレインゲームというジャンルもあります。これは、プレイヤーが、物語の登場人物の一人の役割を演じるというゲームで、主観的にゲームが進行する、という点では、アドベンチャーゲームにも似ています。

しかし、ロールプレイングは、目的に到達するまでの道筋が幾通りもあり、その変化を楽しむのがゲームの面白さとなっていて、目的達成自体はあまり意味をもっていません。

アドベンチャーゲームが、小説を読むことにたとえられるなら、ロールプレイングゲームは小説を書くことにもたとえられるでしょう。

以上のように、アドベンチャーゲームが他のゲームと異なる点は、「確固とした目的を持つ」、「目的を達成することがゲーム内容となる」、「目的に到達する道は、ただ一つしかない」という点にあります。

アドベンチャーゲーム制作にあたっては、これらの点を念頭に置いて、プログラミングしなくてなりません。

アドベンチャーゲームに適した素材選びのチェックポイント

アドベンチャーゲームは「西部劇」である

●「西部劇」こそアドベンチャーゲームの原点

〝宝捜し〟〝犯人捜し〟〝王女救出〟……、目的は何でもいいのですが、単純に言えばアドベンチャーゲームの本質は「西部劇」です。

アドベンチャーゲームの出生地がアメリカだから、という理由だけではありません。余計な心理描写の少ない、白人対インディアンの勧善懲悪スタイル、変化に富んだ場面展開いろいろな小道具の利用……、それら「西部劇」の総体が、パソコンによるアドベンチャーゲームにぴったりなのです。

もちろん今では、西部劇と言えどもこんな単純なものではありません。アメリカの公民権運動などの成果によって、一方的にインディアンを悪者にすることはできにくくなったからです（本当は、白人がインディアンを侵略したのですから当然ですが）。

したがって、いろいろな夾雑物を除いた冒険活劇としての「西部劇」——勧善懲悪スタイル、変化に富んだ場面展開、小道具の多用など——は、アドベンチャーゲームの原点と言ってよいでしょう。幸いアメリカでは、コンピュータの発達以前から、この手の物語が親しまれていました。

●活劇とパソコン

ですから、小説や映画といった今までのメディアから、コンピュータという新しいメディアへ活劇を移し替えるだけで、素晴らしいアドベンチャーゲームが誕生したのではないか、と思うのです。つまり、古くからある活劇と、新しいメディアのコンピュータという両方の材料が揃ったからこそ、アメリカでアドンチャーゲームが誕生し、多くの支持を得ている理由ではないでしょうか。

したがって、アドベンチャーゲームに共通する物の考え方や約束事の底流には、アメリカ人の感性が色濃く反映されていることになります。Infocom 社の「Zork」や「Deadline」などを見ても、アメリカ人独特の感性が感じられます。

本物のアドベンチャーゲームとは？

●アメリカのアドベンチャーゲーム＝本物か、

最近は、日本でもアドベンチャーゲームが数多くつくられるようになりました。超大作から初心者用のものまで様々です。しかし、それらのほとんどは商業用アドベンチャーゲームです。プロのプログラマーやデザイナー、作家たちが、数ヵ月がかりでつくり上げたものです。

そして、彼らが問題にすることは、「いかにアメリカの雰囲気をかもし出すか」であり、「どこまで本物のアドベンチャーゲームに近づけるか」なのです。

果たして、こんなことは可能なのでしょうか？本物のアドベンチャーゲームが「アメリカのもの」だとすれば、我々は、永遠に本物

をつくることなどできません。本物に近づくことはできるでしょうが、それは永遠にニセモノのままです。ニセモノが本物に転化することはできません。

なぜか？その理由は簡単です。我々はアメリカ人には絶対になれないからです。したがって、本物＝「アメリカのもの」はつくることができない、というわけです。本物のアドベンチャーゲームが「アメリカのもの」ならば、我々は本物を諦めるしかありません。

●本物の中身

ですから、「アドベンチャーゲームの本物」という時の中身は、アメリカ的か否かにあるのではなく、パソコンという新しいメディアに適した活劇をどうつくり上げるか、にあるはずです。至極当然なことです。そこのところを、はき違えている人たちが（プロを含めて）いかに多いことか。

巷に氾濫している雑音にまどわされることはありません。先達としてのアメリカのアドベンチャーゲームのいいところを積極的に学べばいいのです。

ですから、アメリカのアドベンチャーゲームに圧倒されたりする必要はまったくありませんし、正統派や邪道などと悩むこともありません。楽屋落ち、独りよがり……自分の好きな世界をパソコンゲームにすればよいのです。これがアドベンチャーゲームづくりの面白さのはずなのです。

●日記としてのアドベンチャーゲーム

アドベンチャーゲームは解く人は楽しいだろうが、つくる人はさぞかしつまらないだろう、とよく質問を受けます。確かに、読む人のいない小説を書いても、つまらないかも知れません。

しかし、ちょっと待ってください。日記をつけたことがある人ならわかると思いますが、誰に見せるでもない文章を書くことも、また楽しいものです。それと同じで、構成をあれこれ一人で想像しながら、ゲームをつくることもアドベンチャーゲームの楽しみ方の１つです。何も人にやってもらうだけが、楽しみではありません。

BASICの初歩しかわからなかった人、自分でプログラムを１本も組んだことがない人でも、アドベンチャーゲームをつくる資格があるし、ゲームづくりを楽しむこともできるのです。プログラムをつくること、それ自体の楽しさを味わえるはずだからです。

●でも、他人に解いてもらうことこそ最高！

そうは言っても、やはり自分で苦労してつくったゲームを、他人に解いてもらうことほど興奮することはありません。あそこに仕掛けたワナに引っかかるなとか、あのヒントは難しいだろうか……なんて想像するだけでワクワクしてきます。この感覚が何と言ってもゲームをつくる側の醍醐味です。

それでは、ゲームづくりの心構えはこれくらいにして、素材捜しの章を始めることにします。

アドベンチャーゲームの素材を捜す

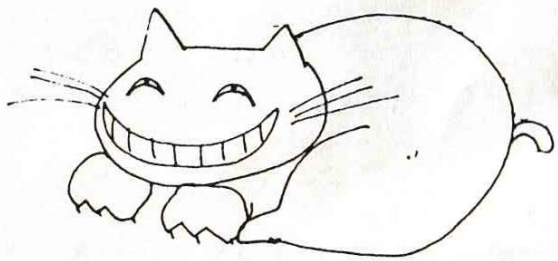
●自分でシナリオをつくる

アドベンチャーゲームをつくる上で、最も苦労するのがこの素材捜しです。ゲームが面白くなるかどうかは、大方素材選びで決まってしまうからです。素材そのものがつまらなければ、いくらプログラミングの高等テクニックを駆使し苦労してつくり上げても、さっぱり面白くならない、という〝悲劇〟的結果に終わってしまいます。

物語をつくるのが大好きでたまらないという人は、自分の書いた物語を素材にゲームをつくればいいのですが、ただちょっと待ってください。自分は面白くても、他人に読んでもらった面白くない場合も往々にしてあります。ですから、プログラミングに取りかかる前に、ストーリーを必ず何人かに読んでもらって、〝客観的〟評価を聞いた上でプログラミングするようお薦めします。

●既にある物語を利用する

反対に、物語なんてとてもつくれないという人も（多くの人はこの範疇に入るでしょう）アドベンチャーゲームづくりの楽しみを放棄する必要はありません。そんな人は、今まで読んだ本の中から、面白いと思ったものを選び出し、アドベンチャーゲームの素材にすればいいのです。



素材選び(つくり)のポイント

●どんな基準があるか？

世の中には面白い物語がたくさんあります。それを利用しない手はありません。ただし、利用する場合、個人で楽しむ分には構いませんが、商業ベースを目的につくる場合は、関係者の了解をとらねばなりませんから注意してください。(著作権は作者の死後50年間は保護される)。

ただ面白いからと言っても、楽にアドベンチャーゲームに仕立てられるものと、ちょっとやそっとではどうにもならない、というものがあります。最初から難しいものに取り組んだら、途中でギブアップするのがオチです。果敢な努力は買いますが、取っかかりは、楽にアドベンチャーゲームになるネタを選んだ方が無難でしょう。

では、アドベンチャーゲーム化しやすいネタかどうかを見分ける基準は何か？その基準がなければ、とにかくやみくもに素材と考えられるものに当たってみる他ありません。

しかし、今まで、いくつかアドベンチャーゲームをつくってきた経験から言うと、アドベンチャーゲーム化しやすいか否かを見分ける〝基準〟が確かにあるのです。普遍的な基準と言えるかどうかわかりませんが、私の体験的基準を皆さんの参考のために、いくつか挙げてみたいと思います。

●心理描写の多いネタは避けよ

素材のジャンルは、SF、推理、童話、冒険……なんでもよいのですが、一般的に心理描写の多い物語は避けた方が無難です。

アドベンチャーゲームは謎解きゲームですから、1つ1つの場面に隠された謎を解きながら、最終目的に辿りつくというスタイルをとります。そうすると、プレイヤーに対する謎解きのヒント（ヒントがなければとっかかりがない）として、心理という抽象的なものを扱うよりも、事件や物といった具体物を表現する方が容易だからです。

それに、心理は主観的なものですから、一義的なヒントにはなりにくい、ということもあります。

また、これは絵入りアドベンチャーの場合の決定的な致命傷として、心理は具体的な絵にならない、ということもあります。

一般的に、大家の作品ほどこの傾向が強いようです。

●小道具の多いものを選べ

これは、前述の説明からもおわかりのように、ヒントをつくりやすいからです。何しろ小道具の多い方が、プレイヤーを飽きさせないよう、いろいろな仕掛を構成しやすいことが理由です。

●場面変化の多いものを選べ

主人公の活躍する舞台は、多ければ多いほどいい。家の中、車の中、海、山……、と活躍する舞台が多いほど、空間的な広がりがあって、プレイヤーを引きつけることができます。

これは、謎解きに対するヒントというよりも、むしろアドベンチャーゲームそのものの進行のリズム感を出すための重要な要素です

これと関連しますが、伏線が張ってある物語は、より劇的な効果を期待できます。

●スジは1人称で展開するものを選べ

これは考えてみれば当然なことです。例えば、探偵を中心に話が展開していたのに、途中から犯人を中心にして物語が進んでいったのでは、プレイヤーの視点が混乱します。どうしても、このようなネタを選びたい時は、探偵か犯人のどちらか一方に焦点を合わせた展開につくり変える必要があります。

●経時変化の少ないものを選べ

これは、超大作などによくあることですが主人公や登場人物が年を取る場合などのことです。ゲームの中に、経時変化の要素を入れるとなると、とても大変なことになります。

●ダイジェスト版は手っ取り早い

テレビのシリーズものの総集編などに相当するものです。これだと、見せ場がぎゅーっと圧縮されていますので、密度の濃いゲームをつくることができます。

●有名な作品は避けよ

面白いからと言って、誰でも知っている有名な作品は、避けた方がいいと思います。と言いますのは、あまりにも知られていて、謎を解く楽しみが半減してしまうからです。

どうしても、つくりたい場合は、物語の構成だけを借りてきて、舞台は全然別のものに変える必要があります。

素材として「不思議の国のアリス」を

●帯に短しタスキに長し

こうやって、既存の物語をふるい分けていくと、100%ピッタリというものはほとんどなくなってきます。

「帯に短しタスキに長し」です。

いずれにせよ、既存の物語はアドベンチャーゲーム用に書かれていませんから、ある程度修正することは覚悟しなければなりません。また、自分で創作する場合は、この基準を念頭においてつくればよいでしょう。

以上の基準から、この連載でつくるアドベンチャーゲームの素材を選んでみましょう。

ルパン、シャーロックホームズ、スーパーマン……、いろいろ思い浮びますが、よく検討すると、以外に心理描写が多かったりして案外つくりにくいことがわかります。

●「不思議の国のアリス」をネタに

そこで、前述の条件を比較的よく満たしているものとして、ルイス・キャロル作の「不思議の国のアリス」を選んでみました。

なお、「不思議の国のアリス」を持っていない方は、東京図書から出ている本をお薦めします(￥980)。これは、アメリカの有名なパズリストであるマーチン・ガードナーの脚注がついていて、アドベンチャーゲーム化する際に、大変参考になります。

各場面の設定と講成法

「不思議の国のアリス」あらすじ

本題に入る前に、「不思議の国のアリス」のあらすじを簡単に説明しておきましょう。

この物語の主人公のアリスは、大変好奇心の強い少女で、ある日、時計を見ながら、あわてて走り去って行く奇妙なウサギを見つけ、そのあとを追いかけていくうちに、地底深く通じる穴の中に落ちてしまいます。アリスが迷いこんだ地底世界では、人間と動物が話し合うことができるのを始めとして、様々な常識はずれなことが起こります。人間の体のサイズを伸ばしたり、縮めたりする食物や、にやにや笑いだけを残して体は消えてしまうチェシャ猫、トランプのカードそっくりの兵士たちや、王様、女王様など、アリスは、この地底の国で、いろいろな不思議な出来事を経験していきます。

ところが、ふと気がついてみると、アリスは土手の上におり、姉の膝の上で眠っていたのでした。不思議な地底の国は、全て春の陽気がもたらした幻であった、というわけです。

このファンタスティックな物語を、アドベンチャーゲームにするには、いったいどうしたらよいのでしょうか？そこで

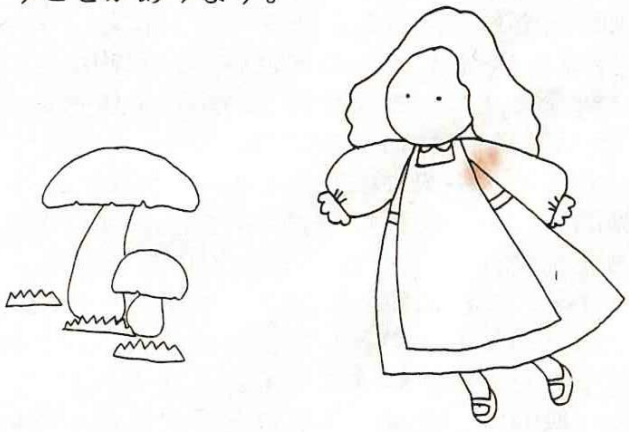
そこで、「不思議の国のアリス」のアドベンチャーゲームストーリーを考えてみることにしましょう。

原作の内容を把握し場面へ分解する

●まず、原作を熟読せよ

「不思議の国のアリス」の内容については、先ほど簡単に触れましたが、実際にアドベンチャーゲームをつくろう！という人なら、原作のストーリーを、完全に把握しておく必要があります。うろ覚えに頼ってゲームのストーリーを組み立てると、前後のつじつまが合わなくなったり、つくった本人だけにしか解けないアドベンチャーゲームになったりしがちです。ですから、アドベンチャーゲームの制作に先立って、最低1回は、原作を初めから終わりまで、通して熟読する必要があります。

自作のストーリーを用いる場合は、物語を最初から最後まできちっと決めておかないと、内容の変更につぐ変更で、話が混乱してしまうことがあります。



●いくつかの段落に分けよ

さて、「不思議の国のアリス」を読み進んでいくと、この物語が、いくつかのまとまりに分けられることに気づくと思います。これは、この物語が最初から小説として考えられたものではなく、作者が子供たち（アリスとその姉妹）に話して聞かせた、即興物語を集めてつくりあげた物語である、ということに由来しているのでしょうが、このことは、アドベンチャーゲーム化するのに、非常に都合がよい構造と言えます。

●各場面に分ける

ご存知のように、アドベンチャーゲームの各場面は、入力要求に対応しています。つまり、アドベンチャーゲームで、入力要求がある時点では、必ず、それに対応する場面が表示されているのです。プレイヤーの条件判断や、ストーリーの分岐などは、全てこの「場面」と、そのつながりの上で成り立っているのですから、アドベンチャーゲームは、たくさんの「場面」が集まって形づくられていると言えます。

「不思議の国のアリス」の物語の構造は、アドベンチャーゲームの構造によく似ているわけです。

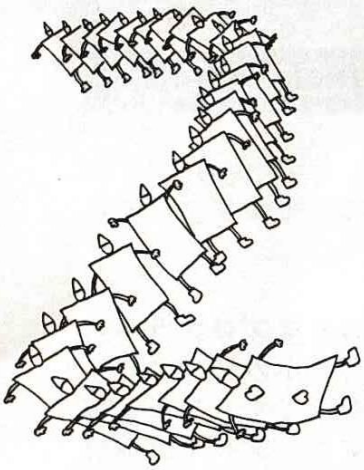
ゲームストーリーを組み立てる

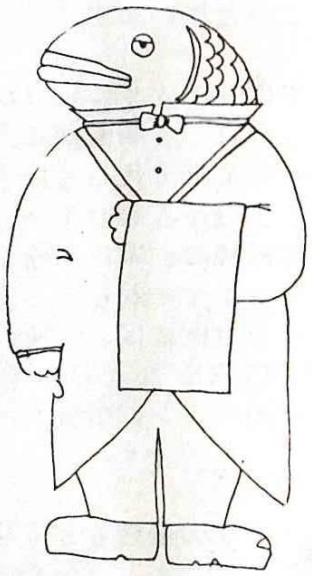
●場面をピックアップする

原作のストーリーをそのままアドベンチャーゲームのストーリーにもっていくことはまず無理でしょう。そこで、各場面を適当に取捨選択し、アレンジを加えて、アドベンチャーゲームのストーリーに適した場面構成に組み立てていく必要があります。取捨選択の基準は、原作選びの時と同様の要領で判断すればよいのです。

「不思議の国のアリス」を例にとれば、涙の池での長い話の場面や、グリフォンといかさま亀との会話の場面などは、会話が多く、動きが少ないので、避けた方がいいでしょう。また、最初の穴に落ちる場面など、物語の「つなぎ」の部分も、アドベンチャーゲームの場面としては、不向きと言えるでしょう。

これらのことを踏まえて取捨選択を行なった後、各場面の内容と相互関係を決めていきます。





●ゲームの面白さはアレンジで決まる

同じ場面構成でも、アレンジによって、ゲームストーリーは全く変えることができます（マイクロキャビンの『ミステリーハウスⅠ』と『ミステリーハウスⅡ』はほとんど同じ部屋割りの家を舞台にしていますが、ストーリーは全然違います。）ゲームの面白さは、アレンジ次第でほとんど決まってしまいます。

場面の設定が原作に忠実なら、相当アレンジしても、原作の雰囲気は損なわずに済みますから、大胆にアレンジして、原作とは全然違うストーリーに仕立ててしまいましょう。そうすれば、原作の雰囲気を味わいながら、謎解きも存分に楽しむことができます（ストーリーまで原作と同じにしてしまっては、原作を読んだことのある人なら簡単に解いてしまうでしょう…）。

●ゲームストーリーをつくる上での注意点

ゲームストーリーをつくる時、注意すべきなのは、１つの場面において、いくつもの分岐、判断を設定することはできるが、１つの分岐、判断の場所として、いくつもの場面を設定することはできない、ということです。先程は、１つしかなかったはずのドアが、あちこち歩いて、元の場所に戻ってきたら、３つに増えていたというのでは、解けなくなってしまいます。

プログラム構造について

それでは「アドベンチャーゲーム」がどんなプログラム構造をもっているかについて書きたいと思います。

●木の枝のように枝分かれするプログラム

これは、各場面があたかも木の枝のように分岐して細分化していく構造です。などと書くと、何やら難しそうですが、プログラム自身は単純で、条件判断と分岐の繰り返しです。

プレイヤーからの入力が、あらかじめ決められた（例えば、〝OPEN DOOR〟〝LOOK PICTURE〟などの）語句と一致すれば、その処理ルーチンに分岐を繰り返して枝分かれしていくものです。

●フラグの処理をどうするか？

この中でネックとなるのは、フラグの処理です。フラグとは、「旗」という意味ですが、コンピュータの世界では、ある状態を示す変数という意味で用いられている用語です。

例えば、あるドアに対応して、変数D1をフラグと決め、そのドアが開けられた状態では、D1は1であり、閉じられた状態では0である、と取り決めておきます。

ゲームスタート時には、そのドアは閉じているものとして、D1は0にしておきます。そして、そのドアを開ける命令が入力された時、その旨の表示を行ない、D1を1とします。これでドアが開いたことになるのです。

フラグを立てる（D1に1を代入する）だけで、どうしてドアが開いたことになるのでしょうか？ドアが開いている時と閉じている時とでは、当然、次の場面に進めるが、進むことができないかの違いがあるはずですが、この処理はしなくてもよいのでしょうか？

実は、これらの条件は、全て条件分岐の時点で処理してしまうのです。つまり、〝ENTER DOOR〟という語句が入力された時、分岐の前に、フラグを検討して、ドアを開いていなければ(D1＝0なら)、この入力は実行できないと処理すればよいのです。その他に、カギがかかっているか否かなど、複数の判断を論理演算を用いて処理します。

●メモリ効効率は悪いが……

さてこの方法では、プログラムの効率はあまり良くありません。例えば、多数のドアがある家を舞台にするなら、その家のドアの数だけの処理ルーチンを必要とし、メモリも沢山使わなければなりません。また、プログラム構造がスパゲティボウル（複雑にからみ合うこと）と化してしまうので、デバッグも困難です。

にもかかわらず、この原始的な処理方式はアドベンチャーゲーム作成上メリットを持っています。

●作業分担は楽

単にプログラミングが易しい、というだけではありません。例えば、何人ものプログラマーが分担してアドベンチャーゲームをつくる時、この枝分かれ構造は、プログラミングのスピードに個人差が出にくく、全く独立にプログラミングを行なうことが可能で、途中変更や結合も容易です。アドリブで全く異なる処理（リアルタイムゲームと結合させたり部分的に効果音を用いたり）を挿入することもできます。

これらのメリットのため、今日でも枝分かれ構造は比較的多用されているいるようです。

データテーブルによる処理

●予想される入力単語をデータテーブルに

これは、アドベンチャーゲームの処理を、全て一つの（または、いくつかの）入出力ルーチンにまとめてしまい、そのパラメータを（入力単語群や出力メッセージ群など）データとして扱う方式です。

プログラムは、まず、入力された語句を、データとして持っていると順次比較、検討し、相当する単語に対してフラグを操作し、出力データ群の中から、その時点の状況に見合うものを選び出して表示します。これらのデータは、一定の規定に従って並べられており、（テーブル化）。プログラムの最後または先頭にまとめて置かせます。

●メモリ効率はよいか

この方式は、メモリ効率も良く、入出力処理が一括して行なえるので、プログラムミスも少なくなります。

データテーブルを持つため、簡単に、どんな単語が使われているかわかってしまうのでは？という問題は、入出力の部分で適当に暗号化を行なえば避けられます。WICSやVTL系の言語、BASICやアセンブラ等、文字列処理に弱い言語でも比較的容易に組めます。もちろん、BASICでもOKです。

しかし、実際には完全に一括処理を行なうことは難しく、部分的に条件分岐を用いている場合が多いようです。

「不思議の国のアリス」の処理

●枝分かれ構造＋データテーブル方式

「不思議の国のアリス」のストーリーを用いて、実際にアドベンチャーゲームをプログラミングするに当たっては、前述した枝分かれ構造と、データテーブル方式の両方を兼ね備えた方式を考えてみました。

このプログラムでは、できるだけ各機種に共通のBASICコマンド・ステートメントのみを使うことを試みました。

出力するメッセージは、配列変数中に文字列の形で書き込み、入力する単語も配列変数に代入して、その中から検索します。これとは別の配列にフラグを圧縮して格納しています。この方法はマシン語プログラミングやポケットコンピュータ用BASICによるアドベンチャーゲームプログラミングの参考になると思います。

●フラグ処理

フラグの内容で、DOORやHOUSE等、同じ単語が複数個使われているのは、1つ1つを区別して表わすためです。例えば、白ウサギと三ヵ月ウサギは共にRABBITですが、全く別のキャラクタとして扱います。

これに対して動詞は場面毎に区別されていません。すべての場面に共通の一単語で動作を表わしています（表2参照）。

移動可能な名詞と移動不可能な名詞も区別してフラグに格納しています。

また、この「不思議の国のアリス」特有なものとして、体の大きさを表わすフラグがありますが、これに類する一種の状態のようなものについても（例えば、眠っている、力が強くなっている。魔法をかけられている等）考慮しなければいけません。しかしごく短期間の状態なら、個別に処理した方が有利なこともあります。

データ圧縮については、いろいろな方法がありますが、ここでは、2進数の各ケタをフラグとして取り出して必要でない時は、10進数にしてしまっておきます。暗号化を考えるなら、素数をいくつか掛けていき、特定の素数で割って、割り切れるか否か確かめるのもよいでしょう。

表1 フラグの内容（名詞）

TH(0,x)　　TH(1,x),,IN

「この数は配列変数の「ケタ」

「この数は名詞ナンバー」

0	BUTLER	27	0	BOTTLE
1	CAT	28	1	FAN
2	CATERPILLAR	29	2	GLOVE
3	CHIMNEY	30	3	HAT
4	CURTAIN	31	4	HATCHET
5	DOG	32	5	KEY
6	DOOR	33	6	MUSHROOM
7	HOUSE	34	7	STICK
8	KING	35	8	PARASOL
9	MARCHIONESS	36	9	ANSWER
10	QUEEN	37	10	HELLO
11	RABBIT	38	11	JUICE
12	SHOP	39	12	MYSELF
13	SMALL DOOR	40	13	OUT
14	STORE	41	14	EYE
15	TABLE			
16	DOOR			
17	DOOR			
18	DOOR			
19	HOUSE			
20	HOUSE			
21	RABIT			
22	SMALL DOOR			
23	TREE			
24	WALL			
25	TREE			
26	TAKO			

表2 動詞のリスト（動詞）

「動詞ナンバー」

1	LOOK	14	OPEN
2	TAKE	15	SAY
3	THROW	16	TALK
4	EAT	17	UNLOCK
5	INVENTORY	18	WEAR
6	GO	19	SAVE
7	BACK	20	LOAD
8	ASK	21	CLIMB
9	DRINK	22	OPEN
10	ENTER	23	JUMP
11	FAN	24	CLOSE
12	GIVE	25	DOWN
13	MOVE	26	FOLLOW

表3 変数リスト

配列 TH(0、x)	シーンxにおいて移動しないもの
配列 TH(1、x)	シーンxにおいて移動するもの（動かせるもの）
H(0)	持っているもの
H(1)	体の大きさ（0…小さい、1…大きい）
CO	動詞、名詞の通し番号（動詞ナンバー＊100＋名詞ナンバー）
FL	移動しない名詞/する名詞のフラグ
配列ＶＲ$	動詞単語群
配列ＮＡ$(1、x)	名詞単語群
配列ＦＭ$	メッセージ文章群

プログラム解説

●初期設定

イニシャライズ後、メインルーチンへ飛びます。ここで、各データの変数への格納、および、各変数の初期設定を行ないます。

各フラグは、変数の中に圧縮されて格納されています。これは主に、メモリー効率を上げるためやガベージコレクションを避けるためです。(ガベージコレクションとは、BASICインタプリタが文字変数領域の整理のために、プログラムが一旦停止することです)。マシン語化への布石ともなるように2進数で表現してあります。

ここで注意すべきは、BASICインタプリタの計算誤差です。この時、機種によっては（PC-8001など)、たとえ2のべき乗の計算でも、桁が大きくなれば一桁以上の誤差を生じるのです。

これを避けて、かつスピードを上げるため、2のべき乗の計算においては、あらかじめその値を計算しておき、配列に代入して使っています。もちろんマシン語を用いればこんな誤差は問題外なのですが……。

●コマンド入力部

コマンド入力の部分です。ここでは、簡単な構文解析しか行なっていませんが、工夫次第では、文章による入力の解析も可能です。

まず、予約してある単語か否かを調べます。「不思議の国のアリス」では、データの暗号化は行なっていませんが、もし行なうとしたら、この部分で翻訳するということになるでしょう。

次は、LOOK、TAKEなど，全てのシーンに共通の動詞の判断です。これらの動詞は、サブルーチン化されて、各々処理しています。

次には、メッセージを表示するサブルーチンがあいてあります。メッセージは変更されることが特に多い箇所なので、まとめて置いておくと便利です。

CHR$(7)はBELLです。「BEEP」が使えない機種で、一番有用なのがこのアスキーコードです。もちろん、音楽演奏や音声出力でも構いません。

●動詞の判断

いよいよ各動詞の判断です。まず、LOOKサブルーチンです。ここで注意してほしいのは、文字型アドベンチャーゲームでは、場面が進む毎に、情景描写を行なうということです。つまり、場面が進む時点で、自動的にLOOKが入力されるということになります。よって、このLOOKサブルーチンは、汎用性に富んでいなくてはなりません。

LOOKの対象となる名詞も情景も、全て変数に代入されていますから、これをシーンNo.から算出して表示します。名詞を伴っている時の場合分けも必要です。また、さらに持ち物の描写を詳しく行ないたい、という人の参考例として、ここでは、持ち物に対してLOOKを行なった場合、「＊carrying＊」というメッセージも同時に出すようにしてみました。

さらに、LOOKして初めて出現する名詞の処理も行なっています（例えば、TABLBをLOOKしたら、その上にKEY が乗っているのを見つけた、という類の処理です)。

この処理は、文字型アドベンチャーでは、うまく構成すれば省略可能と思われますが、情景が大局的に見える、ハイレゾアドベンチャーでは不可欠でしょう。

次は、TAKEサブルーチンです。ここに至って、ようやく固定物と可動物の分離の恩恵にあずかれます。

「不思議の国のアリス」では、「体の大きさ」というパラメータが存在するので、ここでは、その処理（体が小さいと、手がとどかない！）も同時に行なっています。

以上、基本的な部分の説明ですが、結構多くのテクニックを盛り込んでありますので、わかりにくい箇所が多いと思います。しかし、プログラミングのネックとなるのは、フラグの設定と処理の順序などですから、自分がわかりやすい構成でプログラムを組み立ててみてください。そして、行きづまってしまったら、この「不思議の国のアリス」の中に、うまい解決策がないかと、捜してみてください。きっと良い参考となると思います。

ゲームの遊び方

入力は英大小文字、またはカタカナです。入力は、動詞␣名詞の順か、動詞だけです。(英語入力の場合)方向性を持つストーリーではないので、次の面に進みたい時には、「ＧＯ」、戻りたい時には、「BACK」と打ち込みます、（もちろん、すんなり行けるわけではないのですが……)。

また、是非原作の『不思議の国のアリス』を読んでからゲームにとりかかってください。そうすれば、どこをどうアレンジすればアドベンチャーゲームになるか、ということがおのずとわかると思います（できれば、続編、『鏡の国のアリス』も読んでみてください。そして、今後は御自身でつくってみてください……)。

プログラムの改良とその方法

Sceneナンバーはプログラムリスト中のリマーク文に書いてあるので参照して下さい。

プログラム解説

●Scene 1

まず最初に、この場面で有効なコマンドを判別しています。有効でないコマンドに対しても、特別なメッセージを出力したい時にはこの部分で処理してから、再度、入力待ちに戻します。

有効なコマンドに対しては、それに応じた処理を行ないます。この時も主役となるのはフラグ操作です。ある命令が入力された時、これが成立するか否かをフラグによって判断します。

例えばドアを開ける時なら、カギはかかっていないか、体が小さい時は、ドアのノブに手が届かないなどの条件はどうか、すでに開いていないか、などを１つ１つ検討します。もし、条件が満たされない場合は、その旨の表示をしてから、入力待ちに戻します。

この「入力待ちに戻す」処理は、非常に頻繁に行なわれるため、各場面はそれぞれサブルーチンとして取り扱われています。これなら、RETURN 文ひとつでメインルーチンのループに戻すことができます。しかし、この時は、GOSUB 文との対応をしっかりと覚えておいてください。そうしないと、戻るべき場所を見失ってしまいます。プログラミング上達のカギは、ここらあたりの注意力にあるのです。

●Scene 2

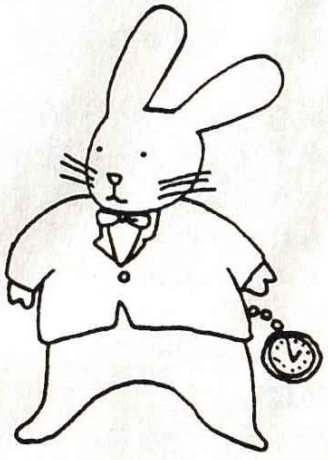
何度も同じコマンドが入力された時、常に変わらぬ表示をするのは、わかりやすい反面面白さに欠けます。というわけで、ここでは、何度もウサギに話しかけると、「何度も話しかけないでくれ……」と答えるように工夫をしてみました。もちろん、「すでにウサギと話し○か否か」のフラグを用いていることは言うまでもありません。

●Scene 3

ゲームが途中で終わってしまう例です。このプログラムでは、途中で終わってしまう理由を説明したのち、END 文で終わってしまいますが、再度ゲームを行なうか否かを尋ねるようにしてもよいでしょう。この時、変数の初期化に気をつけてください。初期化しないと、とんでもないフラグが立ったままになる恐れがあります。CLEAR 文を用いると効果的でしょう。

●Scene 4、5、6、7、8、9、10、11、12、14、15、18

特に書くことはないでしょう。比較的簡単な部分なので、フラグの動きを追いながらプログラムを読んでみてください。



●Scene 13

迷路のような森を抜ける時の方向の入力をここで独立に処理しています。

部分的な処理による入力なので、方向のほかのコマンドは受け付けなくなってしまいますが、工夫次第では解決できると思います。

そのためにはコマンド入力部分がサブルーチン化されていなければなりません。このサブルーチンをコールすることにより、どんな場面においても入力の判断を行なうことができるのです。

ただし、くれぐれもGOSUB RETURNの対応には気をつけてください。ヘタをすると、自分自身をコールしてしまった（再帰的呼出しはBASICでは許されていません）というデバッグの困難なバグを生じることになります。

ただ「不思議の国のアリス」では、コマンド入力それ自体がメインルーチンとなり、各場面処理などをサブルーチンコールする、という構成のため入力の判断を行なうのは困難です。

●Scene 16、17、19

このあたり、体の大きさの変化に気をつけて読んでみてください。

体の大きさ、というような即物的でないものは、持物のリストを取っても表示されませんから、表現を工夫して、プレイヤーにうまく悟らせる必要があります。

●Scene 20

いよいよクライマックスです。このプログラムでは、終わりに近づく程、出力メッセージが長くなっていますが、「ここまでつくればできたも同然」というわけで、プログラマーも思わずいろいろと書きたくなるものなのでしょう。

沢山の表示を一度に行なう時には、途中で一担停止させてプレイヤーに読むヒマを与えます。ゆっくり、徐々に表示していくという方法も考えられますが、ディスプレイ文字を読むスピードの個人差のため、あまり、おすすめできません。

●最終場面処理

アドベンチャーゲームには、必らず目的とするものがあり、エンディングまで到達したということは、当然その目的を果たした、ということになります。

そこで、エンディングメッセージは、祝福の言葉となるのが普通です。しかし、あまりにあっけなく、「おめでとう、おわりです」では、せっかくここまで来た努力も報いられません。美辞麗句を並べたてるのもよいですが、あっ！と驚く意外な結末というのも楽しくて良いかも知れません。

推理ものなら、意外なドンデン返し、というわけです。（「不思議の国のアリス」は、何と「目ざめてみれば夢だった！」という陳腐なエンディングですが、アリスが書かれた当時としては画期的なアイデアだったのでしょう……）。

データのセーブ・ロードを行なう

DATAのセーブ、ロードルチンでは、ゲーム中に変更される可能性のある変数を全てセーブ、ロードしています。具体的には、H(0)、～H(19)、TH(0、X)、TH(I、X)、TR$(0、I)（Xは1～15、Iは0～22まで）です（プログラム参照）。

データのセーブ、ロードは、それぞれの機種によって書式が違いますから注意してください。また、BASIC起動時に「How many files？」と尋ねてくる機種に関しては、ファイル数に1を指定してください。